

---

# NSAnimatablePropertyContainer Protocol Reference

Graphics & Animation: Animation



2009-05-06



Apple Inc.  
© 2009 Apple Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.  
1 Infinite Loop  
Cupertino, CA 95014  
408-996-1010

Apple, the Apple logo, Cocoa, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY**

**DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

## **NSAnimatablePropertyContainer Protocol Reference 5**

---

Overview	5
Tasks	5
Getting the Animator Proxy	5
Managing Animations for Properties	6
Class Methods	6
defaultAnimationForKey:	6
Instance Methods	7
animationForKey:	7
animations	7
animator	8
setAnimations:	9
Constants	9
Transition Animation Keys	9

## **Document Revision History 11**

---



# NSAnimatablePropertyContainer Protocol Reference

---

<b>Adopted by</b>	NSWindow NSView
<b>Framework</b>	/System/Library/Frameworks/AppKit.framework
<b>Availability</b>	Available in Mac OS X v10.5 and later.
<b>Declared in</b>	NSAnimation.h

## Overview

The `NSAnimatablePropertyContainer` protocol defines a way to add animation to an existing class with a minimum of API impact. It returns a proxy object for the receiver that can be used to initiate implied animation of property changes. An object's animator proxy should be treated as if it was the object itself, and may be passed to any code that accepts the object as a parameter. Sending of key-value-coding compliant "set" messages to the proxy will trigger animation for automatically animated properties of its target object.

An object's automatically animated properties are those for which `NSAnimatablePropertyContainer` (page 5) finds and returns an `CAAnimation` instead of `nil`, often because `animator` (page 8) specifies a default animation for the key.

It's perfectly valid to set a new value for a property for which an animation that is currently in progress; this simply sets a new target value for that property, with animation to the new target proceeding from whatever current value the property has reached. An in-flight property animation can be stopped by setting a new value for the property bracketed by an `NSAnimationContext` with `0.0` as the duration.

## Tasks

### Getting the Animator Proxy

- `animator` (page 8) *required method*

Returns a proxy object for the receiver that can be used to initiate implied animation for property changes. (required)

## Managing Animations for Properties

- [animations](#) (page 7) *required method*  
Returns the optional dictionary that maps event trigger keys to animation objects. (required)
- [setAnimations:](#) (page 9) *required method*  
Sets the option dictionary that maps event trigger keys to animation objects. (required)
- [animationForKey:](#) (page 7) *required method*  
Returns the animation that should be performed for the specified key. (required)
- + [defaultAnimationForKey:](#) (page 6) *required method*  
Returns the default animation that should be performed for the specified key. (required)

## Class Methods

### defaultAnimationForKey:

Returns the default animation that should be performed for the specified key. (required)

```
+ (id)defaultAnimationForKey:(NSString *)key
```

#### Parameters

*key*

The action name or property specified as a string.

#### Return Value

The animation to perform. A subclass of `CAAnimation`.

#### Discussion

The [NSAnimatablePropertyContainer](#) (page 5) method consults this class method when its search of the receivers “[Getting the Animator Proxy](#)” (page 5) dictionary fails to return an animation for *key*.

An animatable property container should implement this method to return a default animation to be performed for each key that it wants to make auto-animatable, where *key* usually references a property of the receiver, but can also specify a special animation trigger ([NSAnimationTriggerOrderIn](#) (page 9) or [NSAnimationTriggerOrderOut](#) (page 9)).

A developer implementing a custom view subclass, can enable automatic animation for properties by overriding this method, and having it return the desired default `CAAnimation` subclass to use for each of the property keys of interest. The override should defer to super for any keys it doesn't specifically handle, facilitating inheritance of default animation specifications. The following is an example of such an implementation.

```
@implementation MyView
+ (id)defaultAnimationForKey:(NSString *)key {
    if ([key isEqualToString:@"borderColor"]) {
        // By default, animate border color changes with simple linear
        interpolation to the new color value.
        return [CABasicAnimation animation];
    } else {
        // Defer to super's implementation for any keys we don't specifically
        handle.
    }
}
```

```

        return [super defaultAnimationForKey:key];
    }
}
@end

```

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

NSAnimation.h

## Instance Methods

### animationForKey:

Returns the animation that should be performed for the specified key. (required)

```
- (id)animationForKey:(NSString *)key
```

**Parameters**

*key*

The action name or property specified as a string.

**Return Value**

The animation to perform. A subclass of `CAAnimation`.

**Discussion**

When the action specified by *key* is triggered for an object, this method is consulted to find the animation, if any, that should be performed in response.

Like its Core Animation `CALayer` counterpart, `animationForKey:`, this method is a funnel point that defines the order in which the search for an animation proceeds. It first checks the receiver's “[Getting the Animator Proxy](#)” (page 5) dictionary for a value matching *key*, then falls back to `animator` (page 8) for the receiver's class.

Subclasses should not typically need to override this method.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- [animator](#) (page 8)

“[Managing Animations for Properties](#)” (page 6)

“[Getting the Animator Proxy](#)” (page 5)

**Declared In**

NSAnimation.h

### animations

Returns the optional dictionary that maps event trigger keys to animation objects. (required)

- (NSDictionary \*)animations

### Return Value

The animations as a dictionary.

### Discussion

When an action occurs that may trigger an animation the [NSAnimatablePropertyContainer](#) (page 5) method first looks in this dictionary for an animation that matches the key. Typically, the key will name a property of the object whose value has just changed, but it may specify a special event trigger ([NSAnimationTriggerOrderIn](#) (page 9) or [NSAnimationTriggerOrderOut](#) (page 9)).

### Availability

Available in Mac OS X v10.5 and later.

### See Also

- [animator](#) (page 8)

“Managing Animations for Properties” (page 6)

### Declared In

NSAnimation.h

## animator

Returns a proxy object for the receiver that can be used to initiate implied animation for property changes. (required)

- (id)animator

### Return Value

Returns a proxy object for the receiver that can initiate implied animations in response to property changes.

### Discussion

The animator proxy object should be treated as if it was the receiver itself, and may be passed to any code that accepts the receiver as a parameter.

Sending key-value coding compliant “set” messages to the proxy will trigger animation for automatically animated properties of its target object, if the active [NSAnimationContext](#) in the current thread has a duration value greater than zero, and an animation for the property key is found by the [NSAnimatablePropertyContainer](#) (page 5) search mechanism.

### Availability

Available in Mac OS X v10.5 and later.

### Related Sample Code

BasicCocoaAnimations

CocoaSlides

LayerBackedOpenGLView

UIElementInspector

### Declared In

NSAnimation.h



## setAnimations:

Sets the option dictionary that maps event trigger keys to animation objects. (required)

```
- (void)setAnimations:(NSDictionary *)dict
```

### Parameters

*dict*

A dictionary containing the event trigger keys and associated animation objects.

### Availability

Available in Mac OS X v10.5 and later.

### See Also

[NSAnimatablePropertyContainer](#) (page 5)

- [animator](#) (page 8)

“[Getting the Animator Proxy](#)” (page 5)

### Declared In

NSAnimation.h

## Constants

### Transition Animation Keys

The following constants define the keys that reference the transitions used as views are made visible and hidden.

```
NSString *NSAnimationTriggerOrderIn;
NSString *NSAnimationTriggerOrderOut;
```

#### Constants

NSAnimationTriggerOrderIn

The key that references the transition animation used when a view becomes visible, either as a result of being inserted into the visible view hierarchy or as a result of the view no longer being set as hidden.

Available in Mac OS X v10.5 and later.

Declared in NSAnimation.h.

NSAnimationTriggerOrderOut

The key that references the transition animation used when a view is no longer visible, either as a result of being removed from the visible view hierarchy or as a result of the view set as hidden.

Available in Mac OS X v10.5 and later.

Declared in NSAnimation.h.



# Document Revision History

---

This table describes the changes to *NSAnimatablePropertyContainer Protocol Reference*.

Date	Notes
2009-05-06	Corrected typos.
2007-10-31	Corrected minor typos.
2007-07-24	New document that describes the protocol used to provide an animatable proxy object for views and windows.

**REVISION HISTORY**

Document Revision History